Time Varying Dynamic Bayesian Networks

Richard Suhendra

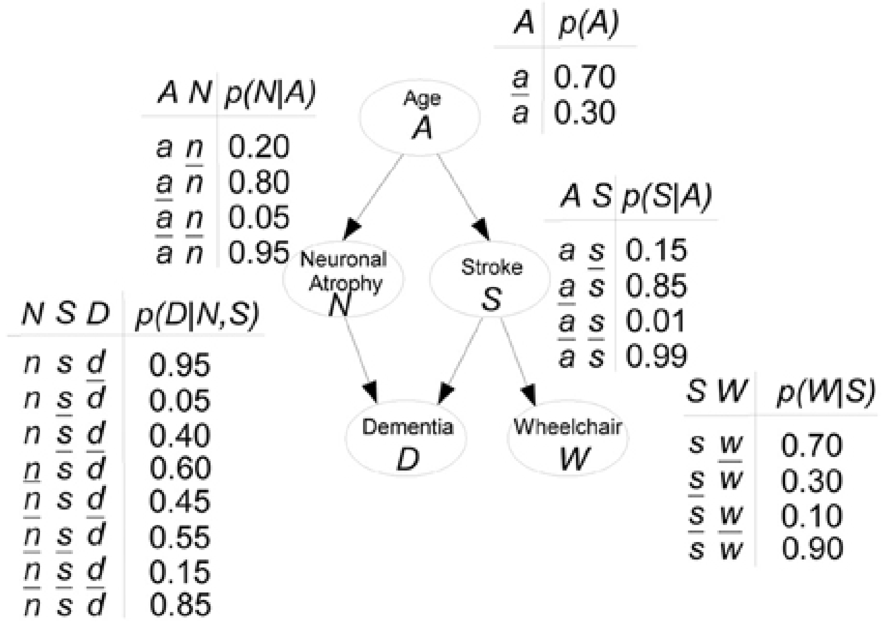
March 2022

1 Introduction

In this project, we briefly introduce the idea of Bayesian networks. We then extend the idea to dynamic Bayesian networks, a simple example of which are our classic autoregressive (AR) models.
Finally, we extend to time-varying dynamic Bayesian networks (TV-DBN), which come up often in biological networks. Data scarcity issues come up when trying to learn the network structure of these TV-DBNs.
Imposing some key assumptions allows us to circumvent these issues, and leads to a way to learn these network structures. We then test our implementation on some test models, as well as what happens when we break these key assumptions.

2 Bayesian Networks

Bayesian networks (BNs) are a type of probabilistic graphical model that essentially models its variables with nodes and represent conditional relationships using directed edges.



A	$p(A)$
a	0.70
$\bar{a}$	0.30

A	N	$p(N\|A)$
a	n	0.20
a	$\bar{n}$	0.80
$\bar{a}$	n	0.05
$\bar{a}$	$\bar{n}$	0.95

A	S	$p(S\|A)$
a	s	0.15
a	$\bar{s}$	0.85
$\bar{a}$	s	0.01
$\bar{a}$	$\bar{s}$	0.99

N	S	D	$p(D\|N,S)$
n	s	d	0.95
n	s	$\bar{d}$	0.05
n	$\bar{s}$	d	0.40
n	$\bar{s}$	$\bar{d}$	0.60
$\bar{n}$	s	d	0.45
$\bar{n}$	s	$\bar{d}$	0.55
$\bar{n}$	$\bar{s}$	d	0.15
$\bar{n}$	$\bar{s}$	$\bar{d}$	0.85

S	W	$p(W\|S)$
s	w	0.70
s	$\bar{w}$	0.30
$\bar{s}$	w	0.10
$\bar{s}$	$\bar{w}$	0.90

Figure 1: Hypothetical example of a BN modeling the risk of dementia. Picture from [1].

Let our BN be $\mathcal{G} = (V, E)$, the following are some important properties:

- Each node corresponds to a random variable $V = \{X_1, ..., X_n\}$ and each edge correspond to a conditional relationship.
- $\mathcal{G}$ has to be directed acyclic graph (DAG).

- $\mathcal{G}$ satisfies the Markov condition: each node is conditionally independent of its non-descendants given its parents. X is c.i of Y given Z if

$$p(x,y|z) = p(x|z)p(y|z) \quad \forall x,y,z$$

or equivalently

$$p(x|y,z) = p(x|z) \quad \forall x,y,z$$

We can use BNs to do inference, or we can use them as models to find causality. As a result of this, a great deal of effort is put into learning the DAG structure of a BN from data, since this can in some sense represent causality. This will be a key detail later.

3 Dynamic Bayesian Networks

Dynamic Bayesian networks (DBNs) are essentially BNs that can be used to model variables that evolve over discrete time slices. We start with some prior network structure. Evolution in time is then modeled by an transition rule that specifies how the time slices are connected from timestep to timestep.
We note that DBNs are stationary models. The transition rule (and therefore the dynamics) stay the same over time.

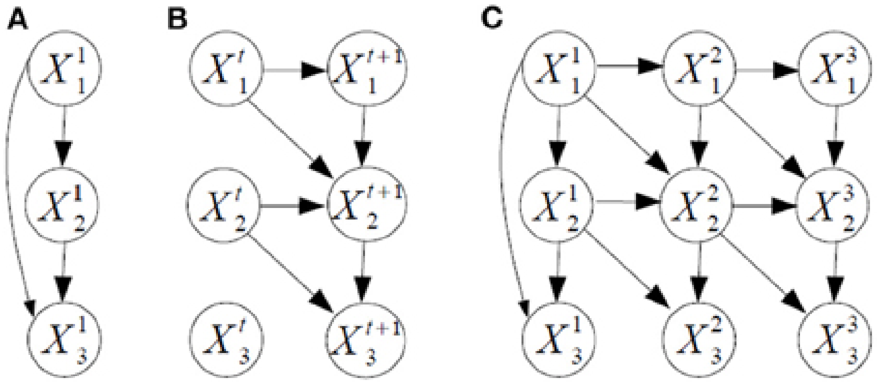


Figure 2: Example of DBN structure with three variables X_1, X_2, and X_3 and three time slices. (A) The prior network. (B) The transition network, with first-order Markov assumption. (C) The DBN unfolded in time for three time slices. Picture from [1].

A simple form of the transition rule is a linear dynamics model:

$$\boldsymbol{X}^t = \boldsymbol{A} \cdot \boldsymbol{X}^{t-1} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}\right) \tag{1}$$

This is in fact our familiar AR(1) model. In fact, AR models are an elegant framework to view our DBNs, since we can simply read off the nonzero entries of the transition matrix $\boldsymbol{A}$ to get the edge structure between time slices $\boldsymbol{X}^{t-1}$ and $\boldsymbol{X}^t$.

4 Time-Varying Dynamic Bayesian Networks and estimating them

Time-Varying Dynamic Bayesian networks (TV-DBNs) [2] are essentially DBNs where the update rule can change over time. In the case of AR models, we have the more generalized form of (1),

$$\boldsymbol{X}^t = \boldsymbol{A}_t \cdot \boldsymbol{X}^{t-1} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}\right) \tag{2}$$

Having a different update rule at each step presents a difficulty if we are interested in learning each of these update rules. Given only one time series, we would be faced with a data scarcity issue, since we would essentially be trying to estimate each matrix from just one transition.
We therefore need to introduce some key assumptions about our transition rules:

1. All matrices are sufficiently sparse
2. Matrices vary smoothly in time

With these two ingredients, the final technique comes by introducing a weighing function in time. Essentially, we use all available data to solve the matrix, but weigh them (for example with a Gaussian RBF kernel) according to how close they lie to the data point of interest.
Since the relevant matrix information for the evolution of each variable lies only the rows of each matrix, our problem can be broken down into an estimation along two orthogonal axes, one in time and one in the space of variables. This gives us

$$\hat{\boldsymbol{A}}_{i\cdot}^{t^*} = \underset{\boldsymbol{A}_{i\cdot}^{t^*} \in R^{1\times n}}{\operatorname{argmin}} \frac{1}{T} \sum_{t=1}^{T} w^{t^*}(t) \left(x_i^t - \boldsymbol{A}_{i\cdot}^{t^*} \boldsymbol{x}^{t-1} \right)^2 + \lambda \left\| \boldsymbol{A}_{i\cdot}^{t^*} \right\|_1 \tag{3}$$

Note that this problem differs slightly from the problem of learning the specific entries of the matrices. In fact, this often doesn't work. Rather, since we are interested in the DAG structure we need only learn the non-zero entries of the matrix.

5 Implementation

Here, we present implementations for learning the time-varying network structures. Firstly, we introduce a general algorithm given access to a LASSO solver proposed in [2] .

Algorithm 1 Procedure for Estimating Time-Varying DBN

```
Input: Time series $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^T\}$, regularization parameter $\lambda$ and kernel parameter $h$.
Output: Time-varying networks $\{\boldsymbol{A}^1, \dots, \boldsymbol{A}^T\}$.
  for $i = 1, ..., p$ do
      for $t^* = 1, ..., T$ do
          Scale time series: $\tilde{x}_i^t \leftarrow \sqrt{w^{t^*}(t)} x_i^t, \tilde{\boldsymbol{x}}^{t-1} \leftarrow \sqrt{w^{t^*}(t)} \boldsymbol{x}^{t-1}$ for $(t = 1 \dots T)$
          Fit a Lasso model with data $\boldsymbol{x}^{t-1}(t = 1 \dots T)$ and target $\tilde{x}_i^t(t = 1 \dots T)$, Lasso parameter $\lambda$.
          $\boldsymbol{A}_i^{t^*} \leftarrow \beta$, the coefficients from our Lasso model
      end for
  end for
```

If a prebuilt Lasso solver is not available to the user, one may instead solve the Lasso problem more directly using the "shooting algorithm".

Algorithm 2 Procedure for Estimating Time-Varying DBN (shooting algorithm)

Input: Time series $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^T\}$, regularization parameter λ, kernel parameter h, and tolerance tol.

Output: Time-varying networks $\left\{\boldsymbol{A}^1, \ldots, \boldsymbol{A}^T\right\}$.

```
  Randomly initialize dummy initial matrix $\boldsymbol{A}^0$.
  for $i = 1, ..., p$ do
      for $t^* = 1, ..., T$ do
          Initialize $\boldsymbol{A}_{i.}^{t^*} \leftarrow \boldsymbol{A}_i^{t^*-1}$
          Scale time series: $\tilde{x}_i^t \leftarrow \sqrt{w^{t^*}(t)} x_i^t, \quad \tilde{\boldsymbol{x}}^{t-1} \leftarrow \sqrt{w^{t^*}(t)} \boldsymbol{x}^{t-1}$ for $(t = 1 \ldots T)$
          do
              $\boldsymbol{A}_{temp} \leftarrow \boldsymbol{A}_{i.}^{t^*}$
              for $j = 1, ..., p$ do
                  Compute: $S_j \leftarrow \frac{2}{T} \sum_{t=1}^T \left( \sum_{k \neq j} \boldsymbol{A}_{ik}^{t^*} \tilde{x}_k^{t-1} - \tilde{x}_i^t \right) \tilde{x}_j^{t-1}, \quad b_j = \frac{2}{T} \sum_{t=1}^T \tilde{x}_j^{t-1} \tilde{x}_j^{t-1}$
                  Update: $\boldsymbol{A}_{ij}^{t^*} \leftarrow (\mathrm{sign}(S_j - \lambda)\lambda - S_j)/b_j$, if $|S_j| > \lambda$, otherwise 0
              end for
          while $||\boldsymbol{A}_{i.}^t - \boldsymbol{A}_{temp}||_\infty > tol$          ▷ Could be any convergence criterion
      end for
  end for
```

6 Test Models

We consider AR models of the form (2). We begin by testing our algorithm on the case where $\boldsymbol{A}_t$ stays constant. We follow that by testing on a time-varying version, involving some anchor matrices and their interpolates. For all systems, we start with $\mathcal{N}(0,1)$ random data and use $\sigma = 0.1$

6.1 Stationary model

Here we let

$$\boldsymbol{A}_t = \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

We simulate with $T = 7$, $h = 2$, and $\lambda = 0.1$. Some of the output is shown below:

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_2
[[0 1 0]
 [0 0 1]
 [1 0 0]]
A_2 estimate
[[ 0.          0.63069727 -0.        ]
 [ 0.         -0.          0.7466979 ]
 [ 0.64891978  0.         -0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_3
[[0 1 0]
 [0 0 1]
 [1 0 0]]
A_3 estimate
[[-0.          0.72620952  0.        ]
 [ 0.          0.          0.65796963]
 [ 0.68842746 -0.         -0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

As we can see, the algorithm reliably recovers the correct non-zero entries for the transition matrix.

6.2 Time-varying model

Here we consider the anchor matrices

$$\boldsymbol{A}_I = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \boldsymbol{A}_I = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \end{bmatrix} \quad \boldsymbol{A}_{III} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

We design our matrices $\boldsymbol{A}_t$ using linear interpolation between these anchor matrices, that is, if we want $\boldsymbol{A}_0 = \boldsymbol{A}_I$ and $\boldsymbol{A}_\tau = \boldsymbol{A}_{II}$, then for $t = 0, .., \tau$ we have

$$\boldsymbol{A}_t = \big(1 - \frac{t}{\tau}\big)\boldsymbol{A}_I + \frac{t}{\tau}\boldsymbol{A}_{II}$$

and similarly for matrices between $\boldsymbol{A}_{II}$ and $\boldsymbol{A}_{III}$.
We simulate with $\tau = 3$, $\lambda = 0.05$, and $h = T/\tau$ so that the bandwidth parameter is scaled to the spacing between anchor matrices. Some of the output is shown below:

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_2
[[0.         1.         0.        ]
 [0.         0.16666667 0.83333333]
 [1.         0.         0.        ]]
A_2 estimate
[[-0.          0.85153424 -0.        ]
 [-0.          0.09895942  0.71701864]
 [ 0.86768824 -0.          0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_3
[[0.         1.         0.        ]
 [0.         0.33333333 0.66666667]
 [1.         0.         0.        ]]
A_3 estimate
[[-0.          0.90659546 -0.        ]
 [ 0.          0.17733492  0.66964455]
 [ 0.80877852  0.          0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Again we can see that the algorithm reliably recovers the correct non-zero entries for the transition matrix. However, this comes with a catch. Just like with regular LASSO, the algorithm is dependent on the value of λ. For example, if we use $\lambda = 0.1$ instead, we instead get

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_2
[[0.         1.         0.        ]
 [0.         0.16666667 0.83333333]
 [1.         0.         0.        ]]
A_2 estimate
[[-0.          0.65915357 -0.        ]
 [-0.          0.          0.62883371]
 [ 0.76716813 -0.          0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_3
[[0.         1.         0.        ]
 [0.         0.33333333 0.66666667]
 [1.         0.         0.        ]]
A_3 estimate
[[-0.          0.77696653 -0.        ]
 [ 0.          0.039898    0.54753785]
 [ 0.67529452  0.          0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

7 Failure Cases

In this section, we explore what happens if we break the assumption of continuity between matrices. Here we let $\boldsymbol{A}_t$ cycle between two matrices. We also pick $h = 2$, $\lambda = 0.05$.

$$\boldsymbol{A}_{2k+1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \boldsymbol{A}_{2k} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Here's some of the output below:

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_3
[[0 1 0]
 [0 0 1]
 [1 0 0]]
A_3 estimate
[[-0.09737459  0.          0.6347378 ]
 [-0.          0.         -0.        ]
 [ 0.65443292 -0.         -0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
A_4
[[0 0 1]
 [0 1 0]
 [1 0 0]]
A_4 estimate
[[-0.         -0.          0.67194651]
 [-0.          0.         -0.        ]
 [ 0.67959288 -0.         -0.        ]]
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

As we can see, the transition matrix recovery does not work well.

8 Discussion and Conclusions

In this project, we presented time-varying Bayesian networks, as well as an algorithm to find these non-stationary network structures. While they can be a powerful tool, there underlies a dependence on λ because of the underlying LASSO solve, as well as an adherence to the assumptions listed above.

References

[1] Concha Bielza and Pedro Larrañaga. Bayesian networks in neuroscience: a survey. *Frontiers in Computational Neuroscience*, 8, 2014.

[2] Le Song, Mladen Kolar, and Eric P. Xing. Time-varying dynamic bayesian networks. In *NIPS*, 2009.